Coding Etiquette for Social Scientists

Marco Morales
mam2519@columbia.edu

GR5069
Topics in Applied Data Science
for Social Scientists

Spring 2017
Columbia University

Housekeeping

- Today:
  - continue algorithm review
  - coding etiquette
  - Team progress report
- next week: enjoy your break!
- week after that: **data challenge due at 6PM**

Coding Etiquette

Some general principles

- **80 characters** should be the maximum length of any line in your code
- use only <- to **assign values** to objects

```
# Good
x <- 8

# Bad
x = 8
8 -> x
```

Coding Etiquette

Some general principles

- improve the readability of your code with **spaces**, though never before a comma

```
#Good
inner_join(ForcesTable, by = c("event.id" = "ID")

#Bad
inner_join(ForcesTable,by=c("event.id"="ID")
```

Coding Etiquette
Some general principles

- use extra spaces to **indent and align** your code to enhance readability

```
ForcesTable.Confrontations <- read_excel(
                                    inFileName3,
                                    sheet = 1,
                                    na = "9999"
                                   )
```

- **never mix spaces and tabs** to indent your code

Coding Etiquette

Some general principles

- in general, **do not use names of existing functions or variables** for your new objects

```
# Bad
mean <- function(x) median(x)
TRUE <- 0
FALSE <- T
```

- always start your comments with **# followed by a space**

Coding Etiquette

Some general principles

- name objects consistently - and meaningfully - throughout your scripts
  - objects should always be **lowercase**
  - be consistent if you use **CamelCase**
  - use _ to **separate words**
  - be careful when using `.` (may cause problems with S3)

```
# Good
navy_deaths
NavyDeaths
navy.deaths # use with care

# Bad
navydeaths
ndths
```

Coding Etiquette

Some general principles

- if you find an error in your code, correct it where it happened
  - do not try to fix it from a later chunk of code

Coding Etiquette

Create structured scripts

- each script should perform **only one task**
  - in particular, always separate data manipulation from data analysis in different scripts
- your code should be **as simple as possible**
  - being clever can - and will! - come back to haunt you when sharing or revisiting code

Coding Etiquette

Create structured scripts

- start your script with a section that provides **all relevant information** that may help you and others make sense of it in the future

```
#   ######################################################################
#       File-Name:      MakeGraphs_CongressRollCall_160603.R
#       Version:        R 3.3.1
#       Date:           June 03, 2016
#       Author:         MM
#       Purpose:        Exploratory graphs of congressional roll call
#                       data for the 112th US Congress. Simple initial
#                       visualizations to find patterns and outliers.
#       Input Files:    ProcessedRollCall_160225.csv
#       Output Files:   Graph_RollCall_112Congress.gif
#       Data Output:    NONE
#       Previous files: MakeGraphs_CongressRollCall_160524.R
#       Dependencies:   GatherData_CongressRollCall_160222.R
#       Required by:    NONE
#       Status:         IN PROGRESS
#       Machine:        personal laptop
#   ######################################################################
```

Coding Etiquette

Create structured scripts

- create a section in your script where you define **important objects** that will be used throughout the code, instead of adding them manually in the code

```
# ::::::: SOME USEFUL DEFINITIONS :::::::::::::::::::::::::::::::::::::::::::::

# set the general path for the project at its root, specific files will define
# their own branches individually

path <- "~//Dropbox//GR5069_Spring2017//GR5069//week_06//datachallenge1"

# define additional paths for files you will use. In each case, determine
# appropriate additions to the path

inFileName1  <- "data//raw//A-E.xlsx"          # raw data on confrontations
inFileName2  <- "data//external//ARCH535.csv"  # name equivalence tables
inFileName3  <- "data//raw//tabla9-A-E.xlsx"   # id of federal forces conf
inFileName4  <- "data//raw//A-A.xlsx"          # raw data on agressions
inFileName5  <- "data//raw//tabla9-A-A.xlsx"   # id of federal forces agg
```

Coding Etiquette

Create structured scripts

- each section of your script should perform a **single task**

```
# ::::::::::::::::: APPLY INITIAL DEFINITIONS :::::::::::::::::::::::::::::::::

setwd(path)
getwd()


# :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# ::::::::::::::::::::::: LOADS DATA ::::::::::::::::::::::::::::::::::::::::::

Confrontations <- read_excel(inFileName1,
                             sheet = 1,
                             na = "9999"   # converting sentinel value to null
)


# ::::::::::::::::::::: SOME DATA PROCESSING ::::::::::::::::::::::::::::::::::

Forces.Confrontations <- TableWrangler(ForcesTable.Confrontations, ForcesNameLookup)

Forces.Aggressions <- TableWrangler(ForcesTable.Aggressions, ForcesNameLookup)
```

Coding Etiquette

Comment your code!!

- include **comments before each block of code** describing its purpose

```
# :::::: LOADING NAME CONVERSION TABLE
# the original file treats numeric codes as strings, must convert to integers
# upon loading. Also, names of municipalities are in Spanish, so must specify
# the encoding as the file is read

NameTable <- read_csv(inFileName2,
                      col_types = cols(
                          CVE_ENT = col_integer(),
                          NOM_ENT = col_character(),
                          NOM_ABR = col_character(),
                          CVE_MUN = col_integer(),
                          NOM_MUN = col_character()
                          ),
                      locale = locale(encoding = "ISO-8859-1")
                      )
```

Coding Etiquette

Comment your code!!

- **comment your functions** thoroughly, including inputs and outputs

```
dataMunger <- function(baseEventData, StateNames, ForcesTable, SourceString){

    # ::::::::: DESCRIPTION
    #
    # The function performs the following transformations in the data to
    # produce the desired output data:
    #
    # 1. add actual names of states and municipalities from a Census table;
    #    currently the database only has their numeric codes
    # 2. rename columns from Spanish to English (not everyone speaks both languages)
    # 3. adding a new variable that indicates the armed force involved in the
    #    confrontation event
    # 4. replace all missing values with 0; this will come in handy as we start to
    #    explore the data futher
    #
    # ::::: INPUTS
    #
    # i)   BaseEventData - the raw database to be munged
    # ii)  StatesName - a table with State/Municipality names
    # iii) ForcesTable - a table that identifies armed forces involved in the event
    # iv)  SourceString  - a string that will identify origin of the table
    #
    #:::::: OUTPUT
    #
    # the function returns a dataframe
```

Coding Etiquette

Comment your code!!

- include comments for any line of code **if meaning would be ambiguous** to someone other than yourself
- sometimes not only the **why** but the **what** may be needed for others to understand the code

```
# filling in NAs with zeros, to facilitate graphing and basic computations
# replace_na() requires a list of columns and rules to apply. Code below
# provides that
replace_na(
    # creates an object with numeric column names
    setNames(
        lapply(          # applies a function that links numeric column names
                         # with the asignment of 0
           vector("list", length(select_if(., is.numeric))), # creates list len= 25
           function(x) x <- 0),  # defines assignment of 0 to numeric col names
        names(select_if(., is.numeric)))  # provides numeric column names
)
```

Coding Etiquette

Comment your code!!

- separate your code into distinguishable chunks using visually distinct characters like `:`, −, or =

```
# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# :::::::::::::::::::::: LOADS DATA ::::::::::::::::::::::::::::::::::::::::::::

AllData <- read_csv(inFileName1)


# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

Coding Etiquette

Data transformation: object/variable names

- use object/variable names that have substantive meaning

```
rename(
        detained = DE,
        total.people.dead = PF,
        military.dead = MIF,
        navy.dead = MAF
       )
```

Coding Etiquette

Data transformation: object/variable names

- code each object/variable so that it corresponds as closely as possible to a verbal description of the substantive hypothesis the variable will be used to test

```
rename(
       female = ifelse(wounded change >0, 1, 0)
      )
```

Coding Etiquette

Data transformation: object/variable names

- use object/variable names that indicate direction where possible

```
rename(
       wounded.increase = ifelse(
                              total.wounded.change >0, 1, 0)
      )
```

Coding Etiquette

Data transformation: validation

- verify that transformed variables resemble what you intended

```
# create a new global unique ID
AllData %<>%
+     mutate(
+         global.id = 1:nrow(.)
+     )

# verify there are no duplicates
length(AllData$global.id) == length(unique(AllData$global.id))
[1] TRUE

# a quick look to see the distribution of the variable
summary(AllData$global.id)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1    1350    2698    2698    4047    5396
```

Coding Etiquette

Data transformation: validation

- verify that missing data is handled correctly on any recode or creation of a new variable

```
# computes lethality indices
AllData %<>%
+   mutate(organized.crime.lethality =
+              organized.crime.dead /
+              organized.crime.wounded
+   )

# exploration to identify undefined values
summary(AllData$organized.crime.lethality)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      0       1     Inf     Inf     Inf     Inf    3090
```

Team Progress Review

Coding Etiquette for Social Scientists

Marco Morales
mam2519@columbia.edu

GR5069
Topics in Applied Data Science
for Social Scientists

Spring 2017
Columbia University